APPENDIX D

UHF/DIRECT SEQUENCE MAC-D PROTOCOL SPECIFICATION

Introduction.

This document, in conjunction with the OWL network frame format specification, defines the MAC-D link protocol used on Direct Sequence and synthesized UHF radio links in an OWL network. The document defines peer-to-peer random-access operation for exemplary Direct Sequence and UHF links, and master-slave extensions for UHF links. Peer-to-peer random-access periods alternate with master-slave contention-free periods, if the master-slave extensions are enabled. Random-access operation allows access point and terminal nodes to randomly access the channel whenever the channel is idle. The master-slave extensions assume that a single master controls access to the channel during "contention-free periods". Where possible, master-slave extensions are documented in separate sections, which can be ignored for strictly random-access implementations.

Functional requirements and capabilities.

The MAC-D layer:

- accepts transmit requests from the MAC-R layer and passes frames to the physical layer for transmission.
- filters out frames which do not belong to the OWL network of the local device.
- filters out frames which are not directed to the local device.
- forwards frames to the MAC-R layer which are directly addressed to the local device, or are broadcast or multicast to the local device.
- transparently fragments and reassembles unicast MAC-R PDUs, which exceed the maximum MAC-D frame size.
- retransmits lost (i.e. corrupted) unicast MAC-D frame fragments.
- detects and discards duplicate MAC-D data frames and data frame fragments.
- regulates access to the communications channel.
- maintains and provides diagnostic statistics for higher layers.

Terminology and definitions.

MAC-D PDU - MAC-D sub layer protocol data unit (PDU).

MAC-R PDU - MAC-R sub layer PDU.

frame - a physical layer PDU or the MAC-D PDU contained in a physical layer PDU.

fragment - a MAC-D PDU. "Fragment" is typically used to denote a frame which is part of a sequence of frames which contain the data for a single MAC-R PDU.

A **frame sequence** is a group of 1 or more frame fragments which contain a single MAC-R PDU.

originator - the node which originates the transmission of a frame sequence.

sink - the destination of a unicast frame sequence.

Clear channel detect (CCD) is asserted to indicate that there is not an active transmission within the physical range of the radio.

Busy channel detect (BCD) is the negation of CCD.

A **LBT (listen-before-talk)** non-persistent channel access algorithm is used to acquire the radio channel.

A CA sequence is the transmission of one or more fragments following a single execution of the LBT channel access algorithm. In the absence of errors, a frame sequence is transmitted as a single CA sequence.

An interframe gap is defined as the minimum idle time allowed between frames which are part of a frame sequence.

An interpoll gap is defined as the time between poll frames which belong to a single CA sequence. The maximum interpoll gap time is BUSY_PULSE_TIME.

Physical layer requirements.

The physical layer:

- prefixes synchronization and framing (i.e. length) bytes and appends FCS bytes to transmitted frames.
- removes synchronization bytes, framing bytes and FCS bytes from received frames.
- discards frames with physical layer errors (i.e. FCS errors).

The physical layer must provide the following primitives:

Open - opens the physical port.

Start - turns the receiver on.

Stop - turns the receiver off.

Transmit - transmits a single MAC-D frame fragment as a single physical layer frame. The physical layer is responsible for prefixing synchronization and framing bytes (i.e. length or delimiter) and for appending FCS bytes to the end of the frame.

Receive_indication - posts a frame without physical layer errors to the MAC-D entity. The physical layer is responsible for removing synchronization and framing bytes (i.e. length or delimiter) from the start of the frame, and FCS bytes from the end of the frame. The receive_indication includes a pointer to a MAC-D PDU and the length of the MAC-D PDU.

MAC-D addresses.

Each MAC-D port is assigned a unicast ethernet address. The MAC-R entity in a device registers the ethernet port address and obtains a 2-byte address for each port. Unicast 2-byte addresses are unique within the domain of an OWL LAN ID. Each MAC-D frame (or frame fragment) contains a 2-byte destination address and a 2-byte source address to identify, respectively, the destination and source port of a MAC-D frame.

The high-order bit of a 2-byte MAC-D address is the **multicast bit**. The multicast bit is set ON to denote multicast addresses. The low-order 15 bits are divided into a **node type** (bits 14-13) and a **port ID** (bits 12-0). The node type can be "AP", "terminal", or "any". The default MAC-D address is a multicast address with the respective node type and a port ID of all 1's. An address of all 1's is the MAC-D broadcast address.

If the multicast bit is set ON in either the destination or source address, then the destination node(s) can not respond (i.e. with a POLL or CLEAR frame). A multicast frame sequence consists of a single EOD frame.

MAC-R sub layer interface.

Each node in the network has a single MAC-R entity which invokes a MAC-D entity per port to send and receive frames on the port.

The MAC-D layer must provide the following service primitives. (Note that primitive names and implementation details may change.)

MACD_init - used to initialize the MAC-D entity.

MACD_hl_bind - used to bind a higher layer entity to the MAC-D entity.

MACD_set_LAN_ID - used to set the LAN ID and (optional) LAN ID mask.

MACD_set_address - used to set the MAC-D 2-byte address.

MACD_get_info - used to obtain MAC-D configuration information.

MACD_transmit_request - used to transmit a MAC-R PDU. Transmit requests are sequenced. The MAC-R entity is limited to a single outstanding transmit request until a "transmit_ok" indication is received from the MAC-D layer.

MACD_transmit_error_indication - used to post the status of a transmit request.

MACD_transmit_ok_indication - used to notify the MAC-R entity that a transmit request has been accepted. The MAC-R entity is limited to a single outstanding transmit request. A request is considered "outstanding" until a transmit_ok_indication is received.

MACD_receive_data_indication - used to post a received MAC-R PDU to the MAC-R entity.

MACD_start - used to put the MAC-D entity in a LISTEN state (i.e. turn on the radio receiver).

MACD_stop - used to put the MAC-D entity in an IDLE state (i.e. turn off the radio receiver).

The MAC-R entity initializes each MAC-D entity with MACD_init, binds to the MAC-D entity with MACD_hl_bind, and puts the MAC-D entity in a LISTEN state with MACD_start.

Initially, the MAC-D entity uses a default multicast address consisting of its node type and a port ID of all 1's. The MAC-R entity uses the ethernet address of the MAC-D entity to obtain a unique 2-byte address from an address server in the OWL root node. The MAC-R entity uses MACD_set_address to set the MAC-D address to the unique value. If the root node changes, the MAC-R entity uses MACD_set_address to reset the MAC-D address to the default value, and the process is repeated.

A MAC-D entity uses the MACD_receive_data_indication primitive to post receive MAC-R PDUs to the MAC-R entity. The receive data primitive includes the MAC-D address of the originator, a MAC-D port identifier, and signal strength information.

The MAC-R entity uses MACD_transmit_request to request a MAC-D entity to transmit a MAC-R PDU. If the size of a MAC-R PDU exceeds the maximum MAC-D frame fragment size, then the PDU is transparently fragmented and re-assembled by the MAC-D originator and sink, respectively. The transmit request primitive includes the destination MAC-D address and a 2-bit P_FLAG parameter. If the multicast bit is ON in the destination address, then the frame can not be fragmented. If P_FLAG is non-zero, then

the MAC-D entity delays a random back off time between 0 and P_FLAG_BACKOFF LBT[pflag_value-1] slot times, before transmitting the PDU.

The MAC-R entity can change the MAC-D state from LISTEN to IDLE (i.e. disable the radio receiver) with MACD_stop. The MAC-R entity puts the MAC-D entity back into the LISTEN state with MACD_start.

MAC-D protocol specification.

MAC-D header.

Destination address.

The destination address specfies the multicast or unicast of the destination (i.e. sink) node(s).

Source address.

The source address is the the address of the originator. The source address may be a multicast address if the MAC-D entity has not been assigned a unique unicast port address.

Protocol ID.

The MAC-D entity discards frames which contain a protocol ID value which it does not support. The protocol field is the first field in the MAC-D header and defines the format of the MAC-D header.

LAN ID.

A MAC-D frame belongs to the MAC-R spanning tree specified by the LAN ID in the MAC-D header. The MAC-D entity discards frames which do not belong to the spanning tree to which it is currently attached after, possibly, updating channel reservation information. The MAC-R entity passes the LAN_ID value to the MAC-D entity during initialization.

Channel Reservation.

The channel reservation byte is used to reserve the communications channel for a unicast frame sequence or for a succeeding multicast frame. The channel reservation value is equal to the total number of data bytes in the sequence divided by 16. The reservation value can not be greater than MAX_CHAN_RESERVE. The channel reservation in RFP and DATA frames is echoed in associated POLL frames.

Control byte.

Bit definitions

R/P (request/poll) - the R/P bit is used to distinguish MAC-D request and poll PDUs. The R/P bit is set OFF in request frames. The R/P bit is set ON in a poll frames.

CONTROL - the CONTROL bit is set ON in control request frames, and is set OFF in data (DATA or EOD) request frames. (The CONTROL bit must be zero in poll frames.)

START - the START bit is set ON in the first data (DATA or EOD) fragment in a frame sequence.

STOP - the STOP bit is set ON in the last data (EOD) fragment in a frame sequence.

TYPE(2) - the START/STOP bits are used as frame "type" bits in control request frames and poll frames.

The R/P, CONTROL, and START/STOP (or TYPE) bits define the frame type.

SEQ - the SEQ bit is used to sequence MAC layer data frames, modulo 2. The SEQ bit is used to detect and discard duplicate data fragments. The SEQ bit is 0 in the first data fragment in a sequence. The SEQ bit in a POLL frame is set to the next expected DATA SEQ.

PRIORITY - the PRIORITY bit indicates the priority - high(1) or normal(0) - of a MAC-R PDU. The MAC-R entity associates a priority with each transmit request. The MAC-D entity in a sink passes the priority to the MAC-R entity with each receive data indication. The PRIORITY bit value is the same for all data frames in a sequence.

MODE - the mode bit is set ON for frames sent in master-slave mode; otherwise, it is set OFF.

MAC-D frame types.

The high-order 4 bits in the MAC-D control byte define the frame type. MAC-D PDUs are categorized as either request or poll frame fragments. Request frames are transmitted by an originator and poll frames are transmitted by the sink.

Request frame types.

A DATA frame is used to send higher-layer data.

An EOD (end-of-data) frame is used to send higher-layer data and is the last data frame in a sequence of one or more data frames. Note that a bracket of data frames may consist of a single EOD frame.

An RFP (request-for-poll) frame is used to request polling from a sink.

An ABORT frame is used to abort the transmission of a frame sequence.

An ENQ (inquiry) frame is used to determine the SEQ state of the sink.

An MSP (master slot poll) frame is used by the master to poll child nodes in master-slave mode.*

Poll frame types.

A POLL frame is used to solicit a data frame from the originator and to return the current SEQ state.

A CLEAR frame is used to return the SEQ state and to inform all listening nodes that the last frame in a frame sequence has been received.

A REJECT frame is used to return an undefined SEQ state or to indicate that a received request frame was invalid.

An MSP-ACK frame is sent in response to an MSP frame from the master, in master-slave mode, to request polling by the master.*

* master-slave mode only.

Frame/packet filtering.

When a MAC-D entity is in a LISTEN state it is continuously listening on its assigned port. The MAC-D entity is always in promiscuous mode and receives all MAC-D layer frames. Frames with physical layer (i.e. FCS) errors are discarded. Valid data frames are re-assembled into a complete PDU and are posted to the MAC-R entity if:

1) the protocol ID and LAN ID in the MAC-D header match the MAC-D protocol ID and LAN ID, and

2) the destination address in the MAC header a) is equal to the unicast address of the local port, or b) is an acceptable multicast or broadcast address.

The high-order **multicast bit** is set ON in all multicast or broadcast addresses. A frame with a multicast or broadcast destination address is accepted if the node type specifies a group to which the local node belongs (i.e. AP or any) and either a) the port ID is all 1's, or b) the port ID is equal to the ID of the local port.

The MAC-D entity will simply discard a received frame if the protocol ID in the MAC-D header does not match the MAC-D protocol ID (i.e. assigned at compile time).

The MAC-D entity will update channel reservation information and then discard a received frame if the LAN ID in the header does not match the assigned LAN ID.

If an optional LAN ID mask has been set then received data frames with a broadcast destination address will be posted to the MAC-R entity if the LAN ID in the header ANDed with the mask matches the current MAC-D LAN ID ANDed with the mask. The default LAN ID mask is all 1's (i.e. the LAN ID must match exactly). 0 bits in the mask are used to indicate "don't care" bits. The LAN ID mask allows the MAC-R layer to hear broadcast HELLO frames from other LANs.

A LAN ID of all 1's is used as a universal LAN ID. Received data framess with the universal LAN ID and a broadcast destination address are posted to the MAC-R entity.

Appendix 1 contains example filtering code.

Channel access.

The channel access algorithm is either 1) random-access or 2) master-slave, depending on the current channel access mode.

Random-access mode channel access.

A **listen-before-talk** (LBT) channel access algorithm is executed to gain access to the channel before the first data frame in a CA sequence is transmitted. All other frames in a CA sequence may be sent without executing the channel access algorithm. The idle time between frames which belong to a single CA sequence must be less than the **maximum interframe gap time**. Note that an RFP frame is first in a unicast frame sequence, whereas an RFP or ENQ frame can be first in a CA sequence.

A **CCD slot** is a CSMA slot time which includes <busy sense time> + <turnaround time> + <processing delay>. The **busy sense time** is the elapsed time from the beginning of a transmission until CCD is asserted at a receiver. The **turnaround time** is the time required by a half-duplex transceiver to switch from a receive state to a transmit state. The processing delay includes the hardware/software processing time required to initiate a transmission.

A CA slot is a CSMA slot time and includes the time it takes an originator to transmit an RFP frame plus the time until the sink transmits a response and BCD is asserted at the originator.

An LBT slot is a weighted average of the CA slot time and the CCD slot time, where weighting is based on the expected worst-case number of nodes and the expected worst-case percentage of hidden nodes. LBT back off times are defined as integer multiples of LBT slots.

CCD idle time is the minimum time that a node must sense the radio channel idle before starting a transmission. The CCD idle time is a function of the access priority associated with each frame sequence. The CCD idle time for high priority frames originated by an access point (AP) is 0.

On wired links, a simple CSMA algorithm forces nodes to detect an idle channel for a CSMA idle time, which exceeds the interframe gap time, before initiating a CA sequence.

On radio links "hidden nodes" can cause throughput to be significantly degraded if a simple CSMA algorithm is used for channel access. The LBT algorithm allows nodes to reserve the channel for a CA sequence. The channel reservation in a request frame is echoed in the associated poll frame. Therefore, the reservation will acquiesce any nodes within range of either the originator or sink. Sleeping nodes must detect an idle radio channel for an LBT idle time which exceeds the maximum interpoll gap time, before accessing the channel. By listening for longer than the interpoll gap time, a node will detect a conversation in progress, if either the originator or sink is in range, even if the active transmitter is "hidden".

MAC-D frames contain a reservation field which is used to reserve the channel for the duration of a frame sequence. The reservation in an RFP, ENQ, or DATA frame indicates the number of untransmitted data bytes in the associated frame sequence. A reservation is calculated as the number of untransmitted data bytes in a frame sequence divided by 16. The actual value stored in a unicast fame is limited to MAX_CHAN_RESERVATION, where MAX_CHAN_RESERVATION is greater than one fragment time. The reservation in a POLL or REJECT frame is always equal to the reservation is the associated RFP, DATA, or ENQ frame, with one exception. The reservation in a retransmitted POLL or CLEAR is always set to the original value. The reservation in a unicast EOD frame and CLEAR frame is 0, except for frames "chained" together by an AP. A multicast frame which has a length greater than MULTI_RFP_THRESHOLD bytes, must be preceded by a muticast RFP frame. The reservation in a multicast RFP frame indicates the length of the associated multicast frame and is not limited to MAX_CHAN_RESERVATION (i.e. because the multicast frame is not fragmented). The reservation in a multicast EOD frame can be set to a short non-zero value (i.e. 2) to reduce channel contention for succeeding multicast frames transmitted as a chained group.

If an originator does not receive an expected response (i.e. a POLL or CLEAR) frame from the sink, during the transmission of a frame sequence, then the originator solicits a retransmission of the last response from the sink by sending an ENQ frame to the sink. The reservation in the ENQ frame is the same as the associated data fragment and, therefore, does not include the length of the (unacknowledged) data fragment. If the data (i.e. DATA or EOD) fragment was lost, then the POLL frame for a previous RFP or DATA frame is retransmitted with the original reservation. If the poll (i.e. POLL or CLEAR) frame was lost, then the poll frame for the current data fragment is retransmitted with the current reservation.

The LBT algorithm requires each node to maintain two variables - RESERVE_TIME and RESERVE_NODE. A MAC-D port operates in promiscuous mode and constantly updates channel reservation information. The channel can be reserved by a received frame if 1) the frame is unicast or multicast, 2) the protocol ID matches, and 3) the LAN ID does not match or the destination address does not equal the address of the local port. The RESERVE_TIME variable is set to the current time plus the time required to send the number of bytes specified in the channel reservation field (including overhead) when a unicast frame is received. The RESERVE_NODE variable is set to the concatenated LAN ID and unicast address of the node which reserved the channel - the source of a request frame or the destination of a poll frame. The RESERVE_TIME variable can be increased by any node, but can only be decreased by

the RESERVE_NODE node. A potential sink node must discard an RFP frame if the RESERVE_TIME is greater than the current time and the originator is not the RESERVE_NODE node. A reservation is canceled if a unicast request frame is received from the RESERVE_NODE node and the LAN ID and destination address specify the local port. Therefore, a sink node can respond to an RFP or ENQ (i.e. with a POLL, REJECT, or CLEAR), if, and only if, the channel is not reserved or if the RFP or ENQ was transmitted by the RESERVE_NODE node.

An idle MAC-D entity resets an LBT **retry counter** and senses the channel before initiating a CA sequence. If BCD is asserted or the channel is reserved, then the MAC-D entity increments the LBT retry counter, selects a random **back off** time, and delays for the random back off time before re-accessing the channel; otherwise, the MAC-D entity will start a CA sequence (i.e. by transmitting an RFP frame). If the expected response to an RFP or ENQ frame is lost, then the MAC-D entity will also increment the LBT retry count and delay before attempting to re-access the channel.

The back off time is a random value between 0 and an entry in the TX_BACKOFF_TABLE, indexed by the number of LBT retries. If the channel is reserved, then the remaining reservation time is added to the back off time. A timer is started for the total delay time. The channel is sensed again when the timer expires and the process is repeated. Note that channel reservations are monitored and RESERVE_TIME is updated while the timer is running.

If BCD is not reliable (i.e. can be falsely asserted due to background noise), then BCD should be ignored if the channel does not become reserved within the time required to send a frame of length MULTI_RFP_THRESHOLD bytes.

A node selects a CCD idle time period associated with the priority (high or normal) of a MAC-R PDU for sensing the channel. If the CCD idle time is 0, then the current state of the channel is sensed; otherwise, the channel is sensed busy if BCD is asserted or the channel becomes reserved during the CCD idle time period.

The input which produces BCD is ANDed with a managed boolean variable - BCD_ENABLE. If BCD_ENABLE is FALSE, then BCD is always false. In this case, busy channel assessment is based solely on reservations.

Master-slave mode channel access.

If "master-slave mode" is enabled, then the channel access mode can alternate between peer-to-peer random access and master-slave contention-free access. The channel is considered to be reserved by the master (i.e. access point) during a "contention-free period". All frame sequences are initiated by the master; therefore, the channel reservation and BCD are ignored in master-slave mode. Master-slave mode is described in detail below.

Example random-access transmission sequences.

The following sequences illustrate typical transmission sequences for sending a sequence of frames from an originator to a sink, in random-access mode. The SEQ value for data and poll frames is indicated in parentheses.

Sequence 1 - illustrates a sequence without errors and with no fragmentation:

RFP	----------------->	
	<-----------------	POLL(0)
EOD(0)	----------------->	
	<-----------------	CLEAR(1)

Sequence 2 - illustrates a sequence without errors and with fragmentation:

RFP	----------------->	
	<-----------------	POLL(0)
DATA(0)	----------------->	
	<-----------------	POLL(1)
DATA(1)	----------------->	
	<-----------------	POLL(0)
EOD(0)	----------------->	
	<-----------------	CLEAR(1)

Sequence 3 - illustrates a lost data fragment:

RFP	----------------->	
	<-----------------	POLL(0)
DATA(0)	(lost data fragment)	
ENQ	----------------->	
	<-----------------	POLL(0)
DATA(0)	----------------->	
	<-----------------	POLL(1)
DATA(1)	----------------->	
	<-----------------	POLL(0)
EOD(0)	----------------->	
	<-----------------	CLEAR(1)

In sequence 3, note that the reservation in the retransmitted POLL(0) frame is the same as the reservation in the initial POLL(0) frame.

Sequence 4 - illustrates a lost poll frame:

RFP	----------------->	
	<-----------------	POLL(0)
DATA(0)	----------------->	
	(lost poll frame)	POLL(1)
ENQ	----------------->	
	<-----------------	POLL(1)
DATA(1)	----------------->	

```
                 <-------------------     POLL(0)
EOD(0)           -------------------->
                 <-------------------     CLEAR(1)
```

Sequence 5 - illustrates a reserved channel at the sink:

```
RFP              ------------------->
                                          (channel reserved)
(delay)

RFP              ------------------->
                 <-------------------     POLL(0)
EOD(0)           -------------------->
                 <-------------------     CLEAR(1)
```

Sequence 6 - illustrates a lost EOD frame:

```
RFP              ------------------->
                 <-------------------     POLL(0)
EOD(0)           (lost EOD frame)

ENQ              -------------------->
                 <-------------------     POLL(0)
EOD(0)           ------------------->
                 <-------------------     CLEAR(1)
```

Sequence 7- illustrates a lost CLEAR frame:

```
RFP              ------------------->
                 <-------------------     POLL(0)
EOD(0)           -------------------->
                 (lost clear frame)       CLEAR(1)

ENQ              -------------------->
                 <-------------------     CLEAR(1)
```

Sequence 8 - illustrates a multicast transmission:

```
EOD(0)           -------------------->
```

Sequence 9 - illustrates a multicast transmission which exceeds MULTI_RFP_THRESHOLD bytes:

```
RFP              ------------------->
EOD(0)           ------------------->
```

The multicast RFP frame, in sequence 9, is used to reserve the channel for the duration of the data (EOD) frame.

Sequence 10 - illustrates a unicast transmission where the initial RFP/POLL seqeunce is (optionally) omitted:

```
EOD(0)        ------------------->
             <-------------------        CLEAR(1)
```

Sequence 11 - illustrates a unicast transmission where the initial RFP/POLL seqeunce is (optionally) omitted; the EOD frame is lost; and the sink does not have a receive SEQ table entry for the originator:

```
EOD(0)        (lost EOD frame)

ENQ           ------------------->
             <-------------------        REJECT
EOD(0)        ------------------->
             <-------------------        CLEAR(1)
```

In sequence 11, a channel access is required before the first EOD transmission and the ENQ transmission; however, a channel access is not required for the second EOD transmission.

In general, the LBT algorithm must be used to access the channel before each RFP or ENQ frame is transmitted by the originator. The LBT algorithm must also be used to access the channel before a single-fragment EOD frame is transmitted without a preceding RFP.

Note that the sink must cache the SEQ state of the originator for a time greater than MAX_ENQ_RETRY_TIME after it transmits the CLEAR frame, in case the CLEAR frame is lost. A sink retransmits the CLEAR if it receives an ENQ and the SEQ state of the originator is known; otherwise the sink responds with a REJECT frame.

MAC-D States.

The state of the MAC-D entity depends on the "channel state" and "channel access mode". The channel state can be IDLE, LISTEN, TRANSMIT, or RECEIVE. The channel access mode can be RANDOM-ACCESS or MASTER-SLAVE. If the mode is MASTER-SLAVE, then the master-slave state is OUTBOUND or one of several inbound states; otherwise, the master-slave state is IDLE.

MAC-D operation in the TRANSMIT and RECEIVE states depends on the channel access mode. For master-slave mode, the operation also depends on whether the node is functioning as a master or slave, which is a compile-time option.

High-level MAC-D channel states.

Each MAC-D entity is in one of the following high-level states.

IDLE - The MAC-D entity is not enabled.

LISTEN - The MAC-D entity is enabled and is receiving frames in promiscuous mode. Note that the MAC-D entity can not enter the RECEIVE or TRANSMIT states until it has been assigned a unique unicast port address.

RECEIVE - The MAC-D entity is receiving a unicast frame sequence. The unicast receive state machine is active. In master-slave inbound mode, the RECEIVE state is also entered for slot polling.

TRANSMIT - the MAC-D entity is transmitting a frame sequence. The TRANSMIT type is set to MULTI_TX or UNI_TX, for multicast and unicast transmissions, respectively. If the type is UNI_TX, then the unicast transmit state machine is active.

Note that the MAC-D entity is still receiving frames in promiscuous mode in both the RECEIVE and TRANSMIT states.

Transitions to the RECEIVE and TRANSMIT states can only be made from the LISTEN state. For example, node must complete or abort an active frame sequence before it can transition to/from the RECEIVE state from/to the TRANSMIT state. The unicast transmit and receive state machines cannot be active concurrently.

The MAC-D entity must maintain a transmitter_busy variable to ensure that physical layer transmit requests are single threaded. The transmitter_busy variable is TRUE if, and only if, a physical layer transmission is in progress.

Random-access protocol state machines.

Multicast frames.

No state machine is required to send or receive multicast and broadcast frames. Multicast and broadcast PDUs are transmitted as a single frame when the channel is available. Received multicast or broadcast frames are simply posted to the MAC-R layer or are discarded. Note that a received frame is determined to be multicast if the multicast bit is set ON in either the destination or source address. A multicast frame is always an EOD request frame with both the START bit and the STOP bit set ON. If the size of a multicast frame is more than MULTI_RFP_THRESHOLD bytes, then the orginator must transmit an RFP frame, with a broadcast destination address, to reserve the channel for the duration of the multicast frame.

The reservation field in the RFP frame is set to the number of bytes in the multicast data frame, divided by 16 (i.e. the reservation can be greater than MAX_CHAN_RESERVE).

Frame sequencing.

The state machines below do not include states which reflect the value of send and receive sequence variables - VS and VR. Each data frame (DATA or EOD) is sequenced with a 1 bit SEQ number. The originator stores the value of the SEQ bit for the current frame fragment in VS. The sink stores the value of the next expected SEQ in VR. The originator resets VS to 0 when it sends an RFP frame. The receiver resets VR to 0 when it receives an RFP frame. The sink increments VR (mod 2) each time a data (DATA or EOD) frame is accepted. The originator sets VS to the value of the SEQ bit in a received POLL or CLEAR frame. Note that the SEQ bit in a POLL or CLEAR frame is set to the next expected SEQ value in a data frame. (See the example transmission sequences.) The sink must cache its VR value after it sends a CLEAR frame (i.e. in a receive SEQ state table) so that it can correctly respond to an ENQ frame if the CLEAR frame is lost.

The MAC-D control byte contains a START bit and a STOP bit. The START bit is set on in a first-in-chain (FIC) frame fragment. The STOP bit is set on in a last-in-chain (LIC) frame fragment. The STOP bit is always on in an EOD frame and off in a DATA frame. Both the START and STOP bits are set on in an only-in-chain (OIC) frame. OIC (i.e. EOD) frames are sent in a single fragment.

A MAC-D entity can be implemented so that an OIC EOD frame can be sent without a preceding RFP frame. In this case, the MAC-D entity must also cache its VS value for each active sink node (i.e. in a transmit SEQ state table). Instead of resetting VS to 0, the sink sets VS to the cached value for the current sink before sending the single-fragment EOD frame. A preceding RFP must be transmitted if the size of the frame is greater than RFP_THRESHOLD bytes or if the SEQ value for the sink is not cached. Note that RFP_THRESHOLD should be small (i.e. 0 to 20) to avoid sending larger frames without a preceding channel reservation.

VS is used to store the SEQ value for the originator in the TRANSMIT state machine. VR is used to store the SEQ value for the sink in the RECEIVE state machine. The orginator should delete its transmit SEQ state table entry for the sink when it enters the transmit state machine (i.e. after storing the SEQ value in VS). Likewise, the sink should delete its receive SEQ state table entry for the originator when it enters the receive state machine. The transmit and receive SEQ state table entries are (re)created at the end of a successful frame sequence (i.e. after a CLEAR is received or transmitted, respectively). Therefore, a state table entry will not exist if a frame sequence can not be completed.

Note that the following state tables assume that RFP_THRESHOLD is less than the maximum fragment size; however, the protocol can easily be extended to accomodate an RFP_THRESHOLD value which is greater than the maximum fragment size.

Unicast transmit state machine.

The MAC-D entity starts the unicast transmit state machine to send a unicast sequence of 1 or more frame fragments. Only timer events and unicast poll frames destined to the local port are passed to the transmit state machine.

The MAC-D entity can be implemented so that the RFP/POLL sequence which precedes a unicast transmission is not required for a data frame of less than RFP_THRESHOLD bytes. The second state transition table below descibes the transmit states if the RFP is omitted. An RFP is always required if the originator does not have an entry for the sink in its transmit SEQ state table or if the frame size is greater than RFP_THRESHOLD bytes.

State descriptions:

(LISTEN) - The MAC-D entity does not have an outstanding MAC-R transmit request. The MAC-D state is LISTEN or RECEIVE.

TX_PEND - The MAC-D entity has an outstanding MAC-R transmit request and is waiting to access the channel before entering the transmit state machine. The MAC-D state is LISTEN or RECEIVE.

TX_IDLE_R - Entry point from the TX_PEND state if the PDU size is not less than RFP_THRESHOLD bytes, the sink SEQ state is unknown, or (optionally) an RFP is always required.

RDY_RFP - The state machine has a sequence of 1 or more frame fragments to transmit and is waiting to acquire the channel before sending an RFP.

S_RFP - The state machine has sent an RFP frame and is waiting for a POLL frame.

RDY_ENQ - A DATA frame or the POLL response was lost and the state machine is waiting to acquire the channel before sending an ENQ.

RDY_ENQ_E - An EOD frame or the CLEAR response was lost and the state machine is waiting to acquire the channel before sending an ENQ.

S_ENQ - The state machine has sent an ENQ frame to determine the status of the current DATA frame and is waiting for a POLL frame.

S_ENQ_E - The state machine has sent an ENQ frame to determine the status of the current EOD frame and is waiting for a POLL or CLEAR frame.

S_DATA - The state machine has sent a DATA frame and is waiting for a POLL frame.

S_EOD - The state machine has sent an EOD frame and is waiting for a CLEAR frame.

State descriptions for the optional transmit state table for omitting the initial RFP:

TX_IDLE_NR - (optional) entry point from the TX_PEND state, if the PDU size is less than or equal to RFP_THRESHOLD bytes and the sink SEQ state is known.

S_EOD_NR - The state machine has sent an OIC EOD FRAME, without a preceding RFP frame, and is waiting for a CLEAR frame.

RDY_ENQ_NR - The EOD frame or the CLEAR response was lost and the state machine is waiting to acquire the channel before sending an ENQ.

S_ENQ_NR - The state machine has sent an ENQ frame to determine the status of the current EOD frame and is waiting for a REJECT or CLEAR frame.

Transmit state counters.

The state machine maintains four counters: 1) **retry_count** is used to count the number of consecutive RFP or ENQ frames which are transmitted (i.e. without receiving a response); 2) **data_retry_count** is used to count the number of times that a single data fragment is retransmitted in a CA sequence; 3) **abort_count** is used to count the number of times that a transmission sequence is aborted. An aborted sequence is

restarted at the beginning if abort_count is less than the maximum value; otherwise, an ABORT_RETRY error is returned. 4) lbt_retry_count is incremented whenever the channel is sensed busy or an expected response is not received. It is reset to 0 when a poll frame is received. A random delay is calculated as a function of lbt_retry_count. A CHAN_ACCESS error is returned if lbt_retry_count exceeds the maximum value. (CHAN_ACCESS errors are not included in the state machine logic.)

Transmit state timers.

A POLL_TIMEOUT timer is started following the transmission of an RFP, ENQ, DATA, or EOD frame, when a response is expected. Note that the timer must be started immediately after the associated transmisssion completes. The time-out value is larger than the interframe gap time plus the time required to transmit a POLL or CLEAR frame. If the POLL_TIMEOUT timer expires before an expected response is received, a retry count (i.e. retry_count or data_retry_count) is incremented and a recovery action is initiated, if the retry count has not exceeded the maximum value.

A TX_BACKOFF timer is used to implement a random delay before (re)trying a transmission.

The higher layer can, optionally, specify a transmit timeout in an OMDPI transmit request. If a timeout is specified, then the MAC-D entity starts a TX_TIMEOUT timer per transmit request, when it receives the request. If the timer expires before the transmission is complete, then the transmission is aborted and a TX_TIMEOUT error is returned. The required number of TX_TIMEOUT timers is equal to MAX_TX_REQUESTS, the maximum number of outstanding transmit requests the MAC-D entity can queue.

The state machine must maintain a "current pointer" variable which points to the current frame fragment, in a sequence of frames. The current pointer is advanced if, and only if, a POLL for the next frame in the sequence is received. If more than one transition is specified when a POLL frame is received, the state of the current pointer determines which transition should be taken.

A "delay" function causes a random delay which is a function of lbt_retry_count. The delay function is used when the expected response to an RFP or ENQ frame is lost, or when the expected response to an EOD frame is lost if the EOD was sent without a preceding RFP (i.e. see state S_EOD_NR).

Receive frame events which are not specified should be ignored.

Transmit entry states.

state	event	action	next state
(LISTEN)	a PDU is passed to the MAC-D layer for transmission	reset abort_count; reset lbt_retry_count; execute channel access alogirthm	TX_PEND
TX_PEND	channel free; the length is not less than the RFP threshold or the SEQ state for the destination is unknown; set MAC-D state to TRANSMIT	set MAC-D state to TRANSMIT	TX_IDLE_R
TX_PEND	channel free; the length is less than the RFP threshold and the SEQ state for the destination is known	set MAC-D state to TRANSMIT	TX_IDLE_NR
TX_PEND	channel busy	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	TX_PEND

State table if an RFP is required.

state	event	action	next state
TX_IDLE_R	(an RFP is required)	reset retry_count; reset data_retry_count; set VS number to 0; delete transmit SEQ table entry for sink; send RFP	S_RFP
RDY_RFP	channel acquired	send RFP frame; start POLL_TIMEOUT timer	S_RFP
	channel busy	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_RFP
S_RFP	POLL_TIMEOUT timer expires; max. RFP retries exceeded	return RFP_RETRY error	(LISTEN)
	POLL_TIMEOUT timer expires	increment retry_count; increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_RFP
	POLL(0) received	reset lbt_retry_count; send first DATA frame; start POLL_TIMEOUT timer	S_DATA
	POLL(0) received	reset lbt_retry_count; send single EOD frame; start POLL_TIMEOUT receive timer	S_EOD
S_DATA	POLL_TIMEOUT timer expires	reset retry count; execute channel access algorithm	RDY_ENQ
	POLL received	reset lbt_retry_count; increment VS, mod 2; reset data_retry_count; send next DATA frame; start POLL_TIMEOUT timer	S_DATA
	POLL received	reset lbt_retry_count; increment VS, mod 2; reset data_retry_count; send next EOD frame; start POLL_TIMEOUT timer	S_EOD
S_EOD	POLL_TIMEOUT timer expires	reset retry_count; execute channel access algorithm	RDY_ENQ_E
	CLEAR received; EOD frame accepted	return good	(LISTEN)
RDY_ENQ	Channel acquired	send ENQ; start POLL_TIMEOUT timer	S_ENQ
	Channel busy	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ
S_ENQ	POLL_TIMEOUT timer expires; max. retry_count exceeded	return ENQ_RETRY error	(LISTEN)

	POLL_TIMEOUT timer expires	increment retry count; increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ
	REJECT received; max. abort_count not exceeded	reset lbt_retry_count; increment abort_count	TX_IDLE_R
	REJECT received; max. abort_count exceeded	return ABORT_RETRY error	(LISTEN)
	POLL for next DATA frame received (i.e. POLL was lost)	reset lbt_retry_count; reset data_retry_count; increment VS, mod 2; send next DATA frame; start POLL_TIMEOUT timer	S_DATA
	POLL for next EOD frame received (i.e. POLL was lost)	reset lbt_retry_count; reset data_retry_count; increment VS, mod 2; send next EOD frame; start POLL_TIMEOUT timer	S_EOD
	POLL for current frame received (i.e. DATA frame was lost)	reset lbt_retry_count; increment data_retry_count; sent current DATA frame; start POLL_TIMEOUT timer	S_DATA
	POLL for current frame received (i.e. DATA frame was lost); max. data_retry_count exceeded	return DATA_RETRY error	(LISTEN)
RDY_ENQ_E	Channel acquired	send ENQ; start POLL_TIMEOUT timer	S_ENQ_E
	Channel busy	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ_E
	unicast request frame received	return CLEAR_ABORT error.	(LISTEN)
S_ENQ_E	POLL_TIMEOUT timer expires; max. retry_count exceeded.	return CLEAR_LOST link error to indicate that the MAC-D entity cannot determine if the PDU was received by the sink and that the physical connection may be lost.	(LISTEN)
	POLL_TIMEOUT timer expires	increment retry count; increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ_E
	unicast request frame received	return CLEAR_ABORT error.	(LISTEN)
	REJECT received; max. abort_count not exceeded	reset lbt_retry_count; increment abort count	TX_IDLE_R
	REJECT received; max. abort_count exceeded	return ABORT_RETRY error	(LISTEN)
	POLL received for current EOD frame (i.e. EOD was lost); max. data_retry_count exceeded	return DATA_RETRY error	(LISTEN)
	POLL received for current EOD frame (i.e. EOD was lost).	reset lbt_retry_count; increment data_retry_count; send current EOD frame; start POLL_TIMEOUT timer	S_EOD

	CLEAR received; EOD frame accepted	return good	(LISTEN)

State table if an RFP is not required.

state	event	action	next state
TX_IDLE_NR (optional)	(an RFP is not required)	reset data_retry_count; set VS to transmit SEQ table value for sink; delete transmit SEQ table entry; send OIC EOD	S_EOD_NR
S_EOD_NR	POLL_TIMEOUT timer expires	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ_NR
	CLEAR received; EOD frame accepted	return good	(LISTEN)
	REJECT received	return INVALID_SEQ error	(LISTEN)
RDY_ENQ_NR	Channel acquired	send ENQ; start POLL_TIMEOUT timer	S_ENQ_NR
	Channel busy	increment lbt_retry_count; delay(lbt_retry_count); execute channel access algorithm	RDY_ENQ_NR
	unicast request frame received	return CLEAR_ABORT error	(LISTEN)
S_ENQ_NR	POLL_TIMEOUT timer expires; max. retry_count exceeded	return CLEAR_LOST link error to indicate that the MAC-D entity can not determine if the PDU was received by the sink and that the physical connection may be lost; delete transmit SEQ state table entry for the sink	(LISTEN)
	POLL_TIMEOUT timer expires	increment retry count; increment lbt_retry_count; delay (lbt_retry_count); execute channel access algorithm	RDY_ENQ_NR
	unicast request frame received	return CLEAR_ABORT error	(LISTEN)
	out-of-sequence CLEAR received (i.e. EOD was lost); max. data_retry_count exceeded	return DATA_RETRY error	(LISTEN)
	out-of-sequence CLEAR received (i.e. EOD was lost)	reset lbt_retry_count; increment data_retry_count; resend EOD frame	S_EOD_NR
	REJECT received (i.e. EOD was lost); max. data_retry_count exceeded	return DATA_RETRY error	(LISTEN)
	REJECT received (i.e. EOD was lost).	reset lbt_retry_count; increment data_retry_count; resend EOD frame	S_EOD_NR
	CLEAR received; EOD frame accepted	return good	(LISTEN)
	POLL received	return INVALID_SEQ error	(LISTEN)

Unicast receive state machine.

Only unicast request frames with a destination address equal to the address of the local port address are passed to the unicast receive state machine. Note that unicast frames which do not contain the destination address of the local port are passed to the channel reservation algorithm while the receive state machine is running. Multicast and broadcast frames are simply posted to the MAC-R entity, if a buffer is available, and are not passed to the receive state machine. Note that a frame is determined to be multicast if the multicast bit is set ON in either the destination or source address.

An RX_TIMEOUT timer is started when the MAC-D entity enters the unicast receive state machine. The timer is reset each time a valid request frame is received from the originator of the current unicast frame sequence. The MAC-D entity returns to the LISTEN state if the timer expires. The RX_TIMEOUT value must be greater than the maximum time an originator can take to send a series of ENQ frames to determine the status of a data fragment before aborting a unicast transmission with an ENQ_RETRY error.

Only one frame sequence may be in progress at a time. An ORIGINATOR variable contains the address of the node which originated the current frame sequence. A VR variable contains the current SEQ state of the frame sequence. The receive state machine is started when an RFP frame is received. The ORIGINATOR variable is set to the source address of the RFP frame and VR is set to 0. A sink node will transmit a POLL frame if, and only if, it is in the receive state machine and the source of the associated request frame is the ORIGINATOR node.

Each node must maintain a SEQ state table which contains an entry for each node which has recently originated a valid frame sequence. An entry in the table is discarded when the associated node originates a new frame sequence or an entry can be discarded after it has been in the table for at least an RX_TIMEOUT time period. Note that the table will never have an entry for the current ORIGINATOR node when the receive state machine is running. An entry for the current ORIGINATOR node is added to the table when a CLEAR frame is transmitted and the receive state machine is exited. The entry contains the ORIGINATOR address and the VR value. A MAC-D entity will respond to an ENQ frame with a CLEAR frame if the SEQ state table contains an entry for the source address of the ENQ frame. A MAC-D entity will respond to an ENQ frame with a REJECT frame if an entry for the source address is not in the table and the source address is not the current ORIGINATOR node (i.e. if the receive state machine is running).

If a preceding RFP is not required, the MAC-D entity can respond to an OIC EOD frame, with a CLEAR or REJECTframe, without entering the receive state machine. The receive SEQ state table entry for the originator is updated with the value in a CLEAR frame. The receive SEQ entry is deleted (and a REJECT frame is transmitted) if the EOD frame is not in sequence.

The receiver must reserve enough buffers for a maximum size sequence of frames before sending a POLL frame in response to an RFP frame. Either the entire frame sequence is received successfully or queued fragments from a paritial sequence are discarded after an RX_TIMEOUT time period.

State descriptions:

RX_IDLE - The receiver is not receiving a sequence of unicast frame fragments. The MAC-D entity is not in the RECEIVE state (i.e. it is in the LISTEN or TRANSMIT state).

RX_BUSY - The receiver has sent a POLL frame and is waiting for the next frame in a sequence. The RX_TIMEOUT timer is running and the MAC-D entity is in the RECEIVE state. The MAC-D entity returns to the LISTEN state when the receive state changes to RX_IDLE.

state	event	action	next state
RX_IDLE	RFP received; channel available	start RX_TIMEOUT timer; set ORIGINATOR variable to source address; set VR to 0; delete SEQ state table entry for the originator; send POLL(0)	RX_BUSY
	RFP received; channel reserved	(ignore)	RX_IDLE
	EOD, DATA with START bit set OFF; channel available	delete SEQ state table entry for the source address; send REJECT	RX_IDLE
	EOD, DATA with START bit set OFF; channel not available	delete SEQ state table entry for the source address	RX_IDLE
	(optional) only-in-chain (i.e. START/STOP bits set ON) EOD	set VR to EOD SEQ+1 (mod 2); send CLEAR; update receive SEQ state table entry for originator with VR; post rx PDU to MAC-R; (Note that the receive SEQ state table value is not checked before the frame is accepted.)	RX_IDLE
	ENQ received; channel available; an entry for the originator is in the SEQ state table	Send CLEAR	RX_IDLE
	ENQ received; channel available; no entry for the originator is in the SEQ state table	Send REJECT	RX_IDLE
	ENQ received; channel reserved	(ignore)	RX_IDLE
RX_BUSY	DATA received from active ORIGINATOR; SEQ equals VR	reset RX_TIMEOUT timer; queue fragment; increment VR, mod 2; send POLL (with SEQ equal to VR)	RX_BUSY
	EOD received from active ORIGINATOR; SEQ equals VR	stop RX_TIMEOUT timer; increment VR, mod 2; send CLEAR; add SEQ state table entry for originator; post rx PDU to MAC-R	RX_IDLE
	DATA or EOD received from active ORIGINATOR; SEQ is not equal to VR	(invalid) flush the current sequence	RX_IDLE
	RX_TIMEOUT timer expires	flush the current sequence	RX_IDLE

	only-in-chain EOD received from inactive node	send CLEAR with EOD SEQ+1; update receive SEQ state table entry for originator; post rx PDU to MAC-R; (Note that the receive SEQ state table value is not checked before the frame is accepted.)	RX_BUSY
	not only-in-chain EOD or DATA received from inactive node	delete SEQ state table entry for the source address	RX_BUSY
	RFP received from inactive node	flush the current sequence; reset RX_TIMEOUT timer; delete SEQ state table entry for originator; set ORIGINATOR variable to source address; set VR to 0; send POLL(0)	RX_BUSY
	ENQ received from active ORIGINATOR; channel available	reset RX_TIMEOUT timer; send POLL	RX_BUSY
	ENQ received from active ORIGINATOR; channel reserved	reset RX_TIMEOUT timer; (ignore)	RX_BUSY
	ENQ received from a node which is not the ORIGINATOR; channel available; an entry for the source is in the SEQ state table	Send CLEAR	RX_BUSY
	ENQ received from a node which is not the ORIGINATOR; channel available; no entry for the source is in the SEQ state table	Send REJECT	RX_BUSY
	ENQ received from a node which is not the ORIGINATOR; channel reserved	(ignore)	RX_BUSY

Frame SEQ States.

All unicast MAC data frames are sequenced with a 1-bit sequence number (SEQ). The sequence number is used to detect lost data frames and duplicate data frames.

The MAC-D entity in each node must maintain transmit and receive SEQ state tables for unicast messages. The receive SEQ state table contains an entry for each active MAC-D originator node. The (optional) transmit SEQ state table contains an entry for each active sink node. Each entry consists of a 1-bit SEQ state variable and a network address. Only unicast frames affect state table entries. Receive table entries must be kept for a period longer than the maximum transmit retry time for a single data frame. The

maximum retry time is equal to the maximum ENQ retry time times the maximum number of data retries (i.e. MAX_DATA_RETRY_COUNT). An entry in the transmit SEQ state table can be kept until the space is required for a new entry. If the originator does not have an entry in its transmit SEQ state table for a sink, then the originator must send an RFP frame before sending a data frame. The RFP causes both the originator and sink to, respectively, set the transmit and receive SEQ state to 0. The SEQ state tables are initialized to empty on power-up and whenever the MAC-D address is set.

Receive SEQ states.

SEQ State descriptions:

SEQ_0 - the sink expects the next DATA or EOD packet to have a SEQ number of 0.

SEQ_1 - the sink expects the next DATA or EOD packet to have a SEQ number of 1.

SEQ_UNKNOWN - the sink is not in the receive state machine and does not have an entry in its receive SEQ state table for the originator.

The MAC-D sink caches receive SEQ state variables for active originator nodes. The variable can be set to one of three states listed above. A state of SEQ_UNKNOWN applies to all nodes which do not have entries in the sink's receive SEQ state table. The sink sets the SEQ bit in a POLL or CLEAR frame to denote the next frame that the sink expects. The sink enters the receive state machine when it receives an RFP. The sink maintains a VR variable, while it is in the receive state machine, which contains the current receive SEQ state for the active ORIGINATOR. VR is initialized to 0 and is incremented, modulo 2, each time a data frame is received.

The sink responds to an ENQ as follows: a) a POLL frame, with the SEQ set to VR, is returned if the sink is in the receive state machine and the ENQ is from the active ORIGINATOR; b) a REJECT frame is returned if the ENQ is not from the active ORIGINATOR and the SEQ state is SEQ_UNKNOWN; c) a CLEAR frame is returned if the sink is not in the receive state machine and the SEQ state is SEQ_0 or SEQ_1.

If the sink exits the receive state due to a time out (i.e. a data frame was not received in response to a POLL frame within an RX_TIMEOUT time period) then the receive SEQ state for the sink node is set to SEQ_UNKNOWN. If the sink exits the receive state after sending a CLEAR frame, then the SEQ state for the sink node is equal to the value of the SEQ bit in the CLEAR frame (i.e. the VR value).

The SEQ state of the originator overrides the SEQ state of the sink. If an RFP is not required, the sink always accepts an OIC EOD frame and returns a CLEAR. For example, if a sink receives an OIC EOD, with a SEQ equal to 0, the sink will always return a CLEAR with a SEQ of 1 and set its receive SEQ state to SEQ_1 for the originator.

Transmit SEQ States.

SEQ State descriptions:

SEQ_0 - the originator sends the current data frame with a SEQ number of 0 and expects a POLL or CLEAR with a SEQ number of 1.

SEQ_1 - the originator sends the current data frame with a SEQ number of 1 and expects a POLL or CLEAR with a SEQ number of 0.

SEQ_UNKNOWN - the transmit SEQ state for the sink is unknown and the originator must send an RFP frame to establish the SEQ state.

The originator sets the SEQ bit in transmitted data frame fragments is to the current VS value for the sink. If an RFP is used, the VS value us initialized to 0 before the first data fragment is transmitted. If an RFP is not used, the originator sets VS to the value in the transmit SEQ state table entry for the respective sink node. The state variable can be in one of the three states listed above. The UNKNOWN state applies to all nodes which do not have entries in the originator's transmit SEQ state table. If the state is UNKNOWN, the originator must send an RFP, to set the SEQ state to 0, before sending a data frame. The SEQ state entry for the sink should be deleted when the originator enters the transmit state machine. The VS value for the sink is updated and a stored in a new transmit SEQ state table entry when the CLEAR is received.

A MAC-D entity does not have to maintain a transmit SEQ state table if an RFP is always used. In this case, the transmit SEQ state is always UNKNOWN.

MAC-D Error Codes.

Transmit Request Codes.

ERR_TX_GOOD	0x00	
Interface errors:		
ERR_TX_DISABLED	0x01	- the MAC-D entity is disabled.
ERR_TX_BUSY	0x02	- the maximum number of outstanding transmit requests has exceeded.
ERR_TX_MAX_LENGTH	0x03	- the lenght is greater than the maximum MAC-R PDU size.
Protocol errors:		
ERR_TX_POLL_SEQ	0x11	- the transmission was aborted because a POLL or CLEAR was receive with an invalid sequence.
Link errors:		
ERR_TX_RFP_RETRY	0x21	- the maximum number of RFP retries was exceeded.
ERR_TX_ENQ_RETRY	0x22	- the maximum number of ENQ retries was exceeded.
ERR_TX_DATA_RETRY	0x23	- the maximum number of data retries was exceeded.
ERR_TX_CLEAR_LOST	0x24	- a CLEAR was not received from the sink (i.e. and the maximum number of ENQ retries was exceeded).
Channel access errors:		
ERR_TX_ABORT_RETRY	0x41	- the transmission was aborted and restarted the maximum number of times.
ERR_TX_CHAN_ACCESS	0x42	- the maximum number of channel access retry errors was exceeded.
ERR_TX_CLEAR_ABORT	0x43	- a CLEAR was not received from the sink and the transmission was aborted by a request frame from a third node.

MAC-D constants/variables.

MAX_TX_REQUESTS - 1 to 3, the maximum number of outstanding MAC-R transmit requests that the MAC-D entity can queue.

MAX_FRAG_SIZE = 250 bytes, is the maximum size of a unicast frame fragment.

MAX_SDU_SIZE = 1600 bytes, is the maximum size of a MAC-R PDU.

MAX_PDU_SIZE = 1608 bytes, is the maximum size of a MAC-D PDU.

MAX_MULTI_SIZE = 1600 bytes, is the maximum size of a multicast frame.

DEF_RFP_THRESHOLD - default RFP threshold. Unicast transmit request which exceed the RFP threshold must be transmitted with a preceding RFP.

LBT_SLOT_SIZE is the LBT slot time and is dependent on the bit rate and radio characteristics.

INTERFRAME_GAP is the maximum idle time allowed between frames in a CA sequence and is dependent on the radio type.

BUSY_PULSE_TIME is the maximum interpoll gap time and is a function of the bit rate.

POLL_TIMEOUT is the time an originator waits for an expected poll frame from a sink. The value is equal to the INTERFRAME_GAP time, plus the worst-case time required to send a poll frame.

RX_TIMEOUT is the maximum time that a sink will remain in the unicast receive state machine without receiving a valid request frame in the current frame sequence.

The following values may be different for terminal and AP implementations:

AP constants:

CCD_IDLE_TIME_HIGH = 0, is the CCD idle time for the high(1) priority data.

CCD_IDLE_TIME_NORMAL is the CCD idle time for the normal(0) priority data.

MAX_RFP_RETRY_AP = 20, is the maximum number of times that an originator will send an RFP frame to an AP sink, without receiving a POLL frame, before aborting a frame sequence.

MAX_RFP_RETRY_TERM = 7, is the maximum number of times that an originator will send an RFP frame to a terminal sink, without receiving a POLL frame, before aborting a frame sequence.

MAX_ENQ_RETRY_AP = 10, is the maximum number of times that an originator will send an ENQ frame to an AP sink, without receiving a POLL or CLEAR frame, before aborting a frame sequence.

MAX_ENQ_RETRY_TERM = 7, is the maximum number of times that an originator will send an ENQ frame to a terminal sink, without receiving a POLL or CLEAR frame, before aborting a frame sequence.

MAX_DATA_RETRY_AP = 10, is the maximum number of times that an originator will retransmit a single DATA or EOD frame in a single CA sequence, when the sink is an AP.

MAX_DATA_RETRY_TERM = 7, is the maximum number of times that an originator will retransmit a single DATA or EOD frame in a single CA sequence, when the sink is a terminal.

MAX_ABORT_RETRY = 5, is the maximum number of times that an originator will restart a frame sequence that has aborted by the sink (i.e. with a REJECT).

MAX_LBT_RETRY = 30, is the maximum number of consecutive times that an originator will delay due to a busy or reserved channel before aborting a frame sequence.

TX_BACKOFF_TABLE[lbt_retry_count] = an array of maximum delay values, indexed by the LBT retry count. The actual delay is a random number between 0 and the maximum delay. "lbt_retries" is incremented whenever the expected response to an RFP or ENQ is not received or whenever the channel is sensed busy or reserved before initiating a transmission. "lbt_retries" is initialize to 0 when a transmit request is received and whenever a poll response (i.e. POLL, CLEAR, REJECT) is received from the active sink. Entries in an AP backoff table are lower than the corresponding entries in a terminal backoff table, to prioritize channel access for an AP.

Terminal constants:

CCD_IDLE_TIME_HIGH

CCD_IDLE_TIME_NORMAL

MAX_RFP_RETRY = 10

MAX_ENQ_RETRY_AP = 7

MAX_DATA_RETRY = 7

MAX_ABORT_RETRY = 7

MAX_LBT_RETRY = 30

TX_BACKOFF_TABLE[lbt_retry_count]

Master-slave Extensions.

Contention free periods.

A master-slave extension of the MAC-D protocol allows an AP to, optionally, function as a master for contention-free channel access, in environments with a single AP per coverage area. A master-slave set (MSS) consists of an AP "master" and "slave" nodes, where a slave node is any terminal node attached to the AP. If master-slave mode is enabled then a contention-free period (CFP) is initiated, per MSS, each time a scheduled MAC-R HELLO frame is transmitted by the AP master. The maximum duration of a CFP is less than, or equal to, the inter-HELLO period, and consists of an a) outbound phase, followed by b) a inbound phase. Multicast messages and pending messages are delivered during the outbound phase. Slave nodes are "polled" by the master, during the inbound phase. A CFP is immediately followed by a peer-to-peer random channel access period or the start of the next CFP.

OWL MAC-D provider interface (OMDPI) extensions.

The MAC-R entity in an AP sets the contention-free start (CF-START) flag ON in an OMDPI transmit request, for a HELLO PDU, to indicate the start of the current CFP to the MAC-D entity. The AP uses "chaining" to associate outbound transmissions with the outbound phase of the CFP. (In random-access mode, the MAC-R entity in an AP "chains" transmit requests to reduce channel contention for outbound

transmissions associated with scheduled HELLO packets.) The MAC-R entity sets the CHAIN flag ON, in each OMDPI transmit request, except the last one, which is associated with the transmission of a scheduled HELLO response packet. The MAC-D interface in the AP should allow at least two outstanding transmit requests, to minimize the latency between chained transmissions. The CHAIN flag is set OFF in the last transmit request in a group of chained transmit requests. The MAC-D entity independently initiates the "inbound phase" of the CFP immediately follows the transmission of the last frame in the chained group. (Note that the HELLO frame is the last frame if no multicast or pending messages are associated with the HELLO frame.) The inbound phase ends when the CFP ends. The protocol used during the inbound phase is described in detail below.

Contention-free channel access rules are simple. The master owns the channel for the duration of the CFP and initiates all transmissions without sensing the channel (i.e. BCD and the channel reservation are ignored). A child node may not initiate a transmission during a CFP. A child node responds to the master without sensing the channel, during a CFP.

The master starts a CFP TIMEOUT timer at the start of each CFP. The CFP ends after the CFP_TIMEOUT timer expires, as soon as any active frame sequence is completed. The master broadcasts a CFP_END frame, at the end of the CFP, to mark the end of the CFP and the start of a random access period.

A CFP is established in a child node in two ways. The MODE bit, in the MAC-D header is always set to 1 in any frame transmitted by the master (or a slave) during a CFP. A child node enters a CFP whenever it is in random-access mode and receives a frame, which originated in its MSS, with the MODE bit set to 1.

A child node also maintains a CFP_START timer, and enters a CFP whenever the timer expires. The timer is synchronized to received HELLO frames, and is set to expire at least 1 LBT slot time before the next scheduled HELLO frame. CFP_START timer operation is described in detail below.

The current CFP for a slave node ends a) if it receives a frame from its parent with the MODE bit set to 0, b) at the start of the next CFP, or c) if the node detects an idle channel for CFP_TIMEOUT milliseconds, where CFP_TIMEOUT is slightly greater than twice the BUSY_PULSE_TIME (e.g. 350 milliseconds for UHF). The MAC-D entity in a child node examines the MODE bit in all received frames which originate within the MSS, even if the destination address is for another node. A CFP_TIMEOUT timer is started at the beginning of a CFP and restarted whenever a frame is received with the MODE bit set ON.

A random-access period follows the end of a CFP, if the CFP ends before the start of the next CFP. The MODE bit in the MAC-D header is set to 0 for any frames transmitted during a random-access period. A child node enters random-access mode as soon as it receives a frame from its parent AP with the MODE bit set to 0. In the absence of other traffic, the AP will send a CFP_END frame, which is a CLEAR frame, with a) a channel reservation of 0, b) a MAC-D broadcast destination address, and c) a MODE of 0, to mark the end of the master-slave period and the start of the random access period. A child node can also enter random access mode if it does not receive a frame from its parent AP within CFP_TIMEOUT milliseconds.

A sleeping terminal can wake up and transmit (i.e. in random-access mode) within BUSY_PULSE_TIME milliseconds, in the absence of other traffic.

A child may initiate a random-access transmission at any time during a random-access period, even if the transmission will not complete before the start of the next CFP. Note that the AP sets the offset field in a scheduled HELLO packet to indicate how much the HELLO packet was delayed. If a child node is unable to transmit a queued inbound frame to its parent during the random-access period, it must wait until the next inbound phase.

CFP_START timer operation.

A logicial CFP_START timer, maintained by the MAC-R entity in a slave node, is set to expire at least one LBT slot time before the next scheduled HELLO period. Whenever the timer expires, the MAC-R entity calls MACD_cfp_indicate(), to indicate the start of a new CFP to the MAC-D entity, and immediately restarts the timer with the calculated interval for the next HELLO period. The timer is restarted whenever a HELLO packet is received (i.e. with an adjusted interval) to maintain sychronization with the parent AP. If the timer expires and MAX_HELLO_LOST consecutive HELLO packets have been missed from the parent AP, then the MACD_cfp_indicate() is not called and the timer is not automatically restarted (i.e. the node goes into an unattached state).

The MAC-D entity (re)starts the CFP_TIMEOUT timer whenever MACD_cfp_indicate() is called and whenever it receives a frame with the master/slave mode bit set ON. Note that the MAC-D entity will exit the current CFP when the CFP_TIMEOUT timer expires or when it receives a frame with the mode bit set OFF.

Contention-free polling protocol.

The master-slave MAC-D protocol can provide contention-free polling for up to 200 terminal nodes. (Wireless APs are not supported if contention-free polling is used.) After all multicast and pending frames have been transmitted in the current CFP, the AP transmits a MAC-D master slot poll (MSP) frame to start a master-slave "inbound phase". The time immediately after an MSP transmission is divided into discrete response time slots, where each slot is slightly larger than the time required to transmit an MSP-ACK frame (e.g. 24 milliseconds for UHF). A child node, which has inbound data to send, responds to an MSP frame, with an MSP-ACK frame, in its designated response slot. (OWL MAC-D frame formats are defined in "owlfram.doc".)

An MSP contains a "slot count" field which contains the number of responses slots. A "flags" field in the the MSP contains response mode bits which define the response mode. a) If the response mode is RANDOM, then a slave randomly chooses one of the "slot count" response slots. b) If the response mode is POLL_LIST, then the MSP contains a list of addresses and the "slot count" field contains the number of addresses in the list. Each address corresponds to a response slot. A slave may respond in a slot if its address matches an address in the list. The relative location of the slave's address in the list defines the relative position of its response slot. c) If the response mode is ADAPTIVE, then a slave determines its response slot offset by processing a set of "expansion flags" contained in the MSP.

The MSP "expansion flags" field contains 4 expansion factor bits and a set of bit flags organized into a tree with up to 4 levels, level 0 to level 3. Expansion flag bits and bytes , in an MSP PDU, are numbered from 0 to N in network order (e.g. left to right).

The first bit (i.e. bit 0) defines the expansion factor for level 0. If it is set ON, then level 0 slots are expanded by 4; otherwise, level 0 slots are expanded by 2. Likewise, bits 1 through 3 define the expansion factors for levels 1 through 3, respectively. (Currently, level 3 cannot be expanded.)

The low-order 4 bits of the first expansion flags byte (i.e. bits 4-7) represent four base slots at level 0, numbered from 0 to 3. If each of the 4 bit flags is 0, then the MSP is followed by four response slots. If a level 0 bit flag is set ON, then the base slot is expanded and the sub tree rooted at the bit defines the number of expansion slots derived from the base slot. In general, if a bit is set ON at level j, then the sub tree rooted at the bit defines the number of derived expansion slots. The bits at level 3 are always 0 (and therefore are not included in the MSP). Figure 1 shows the tree for a set of expansion flags with a value of hex. 0D 83 80 04. Note that the expansion factor bits are all 0, therefore the expansion factor for each level is 2.

level 0	1	1	0	1
level 1	1 0	0 0	x x	1 1
level 2	1 0 x x	x x x x	x x x x	0 1 0 0
level 3	00 xx xx xx	xx xx xx xx	xx xx xx xx	xx 00 xx xx

Fig. 1

The "x" bits in the example expansion tree, in figure 1, represent "don't care" bits, and are set to 0. The bits at level 3 are implicitly set to 0. The expansion tree allocates 12 response slots (i.e. 1 for each 0 bit), with 4, 2, 1, and 5 slots allocated, respectively, per sub tree.

The base slot for a slave node is its address, modulo 4, where 0 is the first slot. For example, a slave node with an address of hex. 4007 is assigned to base slot 3. If the expansion flag for the base slot is set ON, the slave node continues to recursively derive a smaller sub tree, rooted at the next level, until the root bit of the derived sub tree is 0. At each level, the node shifts out the previous significant bits of its address and takes its address, modulo the expansion factor, to determine the root bit of the next sub tree. The node responds in the slot which follows any slots allocated by all previous sub trees. For example, in figure 1, a slave node with address hex. 4007 responds in the ninth slot.

A slave node calls **node_slot_offset(nsp, address)**, where "nsp" points to an MSP PDU and "address" is the 16-bit slave address to determine its relative slot number. The function returns a negative value if the node cannot respond in a slot; otherwise, it returns the slot number for the slave node, where 0 is the first response slot. Example code for node_slot_offset() is shown in appendix 3.

The polling algorithm assumes that the AP can detect a collision, if more than 1 node responds in the same slot. If a collision occurs in a slot at level j, then that slot is expanded, by the expansion factor for level j, to level j+1 . If no collisions occur in a set of expanded slots at level j+1, then the slots are compressed to level j.

The total number of response slots can be large, in a heavily loaded network. If the expansion tree defines a large number of slots, then the master limits the number of "active" response slots to a "window" of slots within the total slots. The MSP "slot offset" field contains the offset of the first active slot in the window, and the "slot count" field contains the number of slots in the window. The active slot count is always less than or equal to MAX_RSP_SLOTS (i.e. 8). The master resends the same expansion flags until all of the response slots have been active.

The AP master polls all nodes which successfully respond to an MSP. The master adds the source address, in any received MSP-ACK, to an internal polling list. After the MSP response period ends, the AP sends a "master POLL" frame (i.e. a POLL frame with the MODE bit set to 1) to each node in its list, in priority order. The POLL frame initiates a unicast OWL MAC-D frame sequence. The reservation in the master POLL frame is set to the reservation request value in the corresponding MSP-ACK frame. If a node has inbound data queued it responds to a master POLL frame with a DATA or EOD frame; otherwise, it responds with an ABORT frame. An ABORT causes the node to be deleted from the AP polling list. If a slave responds to a POLL with a DATA frame, then the master immediately sends another POLL frame, to the slave, for the next data fragment. Ifa slave responds with an EOD frame, then the master sends a CLEAR frame to the slave. A slave node is deleted from the polling list whenever the master successfully receives a frame from the slave node (even if the frame is received in random-access mode). If the terminal has more than 1 frame to send, it must wait for the next MSP cycle, or for a random-access period.

An MSP-ACK frame has a "wait time" field which is set to the time that the associated transmit request has been queued, in milliseconds, just before the MSP_ACK is transmitted. The master subtracts the "wait time" value from the current time and enters the difference in the polling list entry for the slave node. Polling list entries are sorted so that the lowest entries are polled first.

After polling each node in its list, the AP sends the next MSP. The inbound phase ends when no MSP-ACKs are received and no collisions are detected in any response slots. A polling cycle is guaranteed to terminate because slots with collisions are not expanded at the highest MSP level. (In the worst case, a slot is allocated per terminal because the maximum number of slots, 256, exceeds the maximum number of terminals.)

In the absence of other outbound traffic, the AP sends an CFP_END frame to mark the end of the CFP (and the inbound phase). Note that it is possible that the inbound phase may not end before the start of the next CFP. In this case, the AP saves its polling list, which contains an entry for each node which successfully responded to an MSP but did not get polled. The AP first polls those nodes in its list before transmitting the first MSP in the next inbound phase.

Initially, a device does not have a unique 2-byte address. A slave node, without a 2-byte address, cannot respond in an MSP response slot. Instead, it must contend for the channel in a random-access period. The master must guarantee periodic random-access on a heavily loaded channel. A master can guarantee a random-access period, for example, by limiting the duration of each CFP, to guarantee a random-access period between successive CFPs, or by skipping every Nth CFP.

Slave state machines.

Slave transmit state machine.

The state transition table below specifies the slave MAC-D protocol operation for the transmission of unicast inbound data in master-slave mode. The slave discards frames which are not from the master therefore only frames from the master are passed to the slave transmit state machine.

Note that the master is responsible for error recovery. If a POLL, DATA, or EOD frame fragment is lost, the master will resend a POLL frame to solicit the (re)transmission of a data fragment. The master checks BCD one CA slot time after sending a POLL, and (re)transmits the next POLL (i.e. to the same node or another node) immediately if BCD is not asserted. The master waits for, at most, BUSY_PULSE_TIME milliseconds, if a response is not received, before sending the next POLL frame. The master limits the number of POLL retries for a single node to MS_POLL_RETRIES (i.e. 3).

In the table below, "other frame" is used to denote any frame other than previously listed frame types for the state. It is assumed that POLL and CLEAR frames are addressed to the station (i.e. POLL or CLEAR frames addressed to another station fall into the category of "other frames").

Note that the current CFP ends if the CFP_TIMEOUT timer expires or if a frame is received from a node in the MSS with the mode bit set to 0. The CFP_TIMEOUT timer is restarted each time a frame is received with the MODE bit set to 1.

State	Event	Action	Next State
(LISTEN)	a PDU is passed to the MAC-D layer for transmission	reset abort_count; set VS number to 0; delete transmit SEQ table entry for sink	MS_TX_PEND
	POLL received	send ABORT	(LISTEN)
MS_TX_PEND	MSP received	calculate slot offset; send MSP_ACK with SEQ equal to VS	MS_POLLING
	CFP ends/random-access period begins		TX_PEND
	POLL received	send ABORT	MS_TX_PEND

MS_POLLING	POLL received; data frame is more than 1 fragment	send first DATA frame fragment	MS_S_DATA
	POLL received; data frame is 1 fragment	send single-fragment EOD frame	MS_S_EOD
	MSP received	calculate slot offset; send MSP_ACK with SEQ equal to VS	MS_POLLING
	CFP ends/random-access period begins		RDY_RFP
MS_S_DATA	POLL received for current DATA fragment	retransmit the current DATA fragment	MS_S_DATA
	POLL received for next DATA fragment	increment VS, mod 2; send next DATA fragment; start MS_POLL_TIMEOUT timer	MS_S_DATA
	POLL received for next EOD fragment	increment VS, mod 2; send last EOD fragment; start MS_POLL_TIMEOUT timer	MS_S_EOD
	MSP received; max. abort_count exceeded	return ABORT_RETRY error	(LISTEN)
	MSP received; max abort_count not exceeded	increment abort count; calculate slot offset; set VS to 0; send MSP_ACK with SEQ equal to VS	MS_POLLING
	other frame received with MODE=1; max. abort_count exceeded	return ABORT_RETRY error	(LISTEN)
	other frame received with MODE=1; max. abort_count not exceeded	Set VS number to 0; increment abort count	MS_TX_PEND
	CFP ends; max. abort_count not exceeded	Set VS number to 0; increment abort_count	TX_PEND
	CFP ends; max. abort_count exceeded	return ABORT_RETRY error	(LISTEN)
MS_S_EOD	CLEAR received; EOD frame accepted	return good	(LISTEN)
	other frame received with MODE=1	return CLEAR_ABORT error	(LISTEN)
	CFP ends	return CLEAR_ABORT error	(LISTEN

Slave receive state machine.

The receive state machine for unicast frame sequences in slave mode is identical to the receive state machine for random-access mode, except that the channel is never busy (i.e. BCD and the channel reservation are ignored).

Master state machines.

Master mode timers.

An MM_RESPONSE timer is started when the master sends a POLL frame and expects to receive a data fragment in response. The timer is started, initially, with a short timeout equal to the CA slot time. If a frame reception is in progress when the timer expires (i.e. if BCD is asserted), the timer is restarted to wait for the end of the transmission; otherwise, the next POLL frame is immediately transmitted.

An MSP_SLOTS timer is started immediately after the transmission of an MSP frame. The interval is equal to the number of MSP response slots times the duration of one slot.

A CFP_TIMEOUT timer can be used to limit the duration of a CFP.

Master transmit state machine

The transmit state machine for unicast frame sequences in master mode is identical to the transmit state machine for random access mode, except that the channel is never busy and a random delay is not used between RFP and ENQ retries.

Master receive state machine.

The state transistion table below describes the operation of the master, in master-slave mode, for receiving inbound unicast frames. The master enters the MM_OUTBOUND state at the start of each CFP and transitions to one of the master-slave master polling (MMP) states at the start of the inbound phase.

The master must maintain a poll list (described above), which contains an entry for slave nodes which have successfully responded to an MSP frame with an MSP_ACK frame. Entries in the list are aged and discarded after MSP_ACK_TIMEOUT seconds. An entry is deleted when the master sends the first POLL to the associated node. An entry is also deleted if a frame is received from the associated node in random-access mode.

There is an immediate transition in the MMP_NEXT state, depending on the state of the poll list and two variables, CFP_start_pending and CFP_end_pending. If CFP_end_pending is TRUE, then the current CFP ends as soon the MAC-D entity completes any active unicast frame sequence. If CPF_start_pending is TRUE, then the next CFP is started as soon as the MAC-D entity completes any active unicast frame sequence. CFP_end_pending is set to TRUE if a CFP is active and either the CFP_TIMEOUT timer expires or the MAC-R entity initiates the next CFP. CFP_start_pending is set to TRUE when the MAC-R entity initiates the next CFP.

If the master MAC-D entity receives a transmit request while in the MMP_S_POLL state, the master can start the outbound transmission as soon as polling is complete for the current node, and then return to the MMP_S_POLL state when the outbound transmission is complete. The transmission is executed according to the unicast transmission state table, shown above, except that the MODE bit is set ON in each frame in the sequence.

State	Event	Action	Next State
MM_OUTBOUND	transmission of the last outbound CF frame completes; polling list is empty	reset collision_count; set poll_list to empty; send initial MSP; start MSP_SLOT(slots) timer	MMP_S_MSP
	transmission of the last outbound CF frame completes; polling_list is not empty	set VR to 0; reset retry_cnt; send master POLL to the first node in the polling list; start MM_RESPONSE	MMP_S_POLL

MMP_S_MSP	MSP_SLOT timer expires; collision_count=0; polling_list is empty	send CFP_END CLEAR frame	(LISTEN) random-access mode
	MSP_SLOT timer expires; collision_count > 0; polling list is empty	adjust slots; send next MSP; start MSP_SLOT(slots) timer	MMP_S_MSP
	MSP_SLOT timer expires; polling_list is not empty	sort poll list by priority; reset retry count; send master POLL to the first node in the polling list	MMP_S_POLL
	MSP_SLOT timer expires; next CFP starts	save polling list	MM_OUTBOUND
MMP_NEXT	CFP_end_pending is TRUE; CPF_start_pending is FALSE	save polling list; send CFP_END frame	(LISTEN) random-access mode
	CFP_end_pending is TRUE; CPF_start_pending is TRUE	save polling list	MM_OUTBOUND
	CFP_end_pending is FALSE; poll list is not empty	set VR to 0; send POLL to the next node in the poll list	MMP_S_POLL
	CFP_end_pending is FALSE; poll list is empty; collision_count > 0	adjust slots; send next MSP; start MSP_SLOT(slots) timer	
	CFP_start_pending is FALSE; poll list is empty; collision_count = 0	send CFP_END (CLEAR) frame	(LISTEN) random-access mode
MMP_S_POLL	receive DATA frame from the polled node	increment VR; reset retry_cnt; send next master POLL to the current node	MMP_S_POLL
	receive EOD frame from the polled node	send CLEAR; save polling list	MMP_NEXT
	receive ABORT frame from the polled node		MMP_NEXT
	MM_RESPONSE timer expires; max. retry count exceeded		MMP_NEXT
	MM_RESPONSE expires; max. retry count not exceeded	increment retry count; resend POLL	MMP_S_POLL
	CFP_TIMEOUT timer expires	set CFP_end_pending flag	MMP_S_POLL
	next CFP start indication is received from the MAC-R layer	set CFP_end_pending flag; set CPF_start_pending flag	MMP_S_POLL

Appendix 1 - Receive frame coding example.

The macd_phy_rx_done "C" routine, shown below, is used to process all received frame events. (Note that the physical layer discards received frames with errors and removes physical layer framing bytes.) The code does not include unicast transmit, receive state machine logic, or master-slave extentsions. A frame is passed to the unicast receive state machine with rx_idle or rx_busy. A frame is passed to the unicast transmit state machine by calling the state function pointed at by cb->tx_state.state.

```
#define MACD_BROADCAST_ADDRESS 0xffff
#define MACD_BROADCAST_LAN_ID    0xff

typedef struct {
  uchar_t protocol_id;
  uchar_t lan_id;
  uchar_t dst[2];
  uchar_t src[2];
  uchar_t ctl;
  uchar_t len;
} macd_header_t;

void macd_phy_rx_done(macd_control_t *cb, o_event_t *evt)
{
  macd_header_t *hdr;
  unsigned len;
  unsigned short dst, src, node_type, port_id;

  /* set the MAC-D PDU (i.e. hdr) pointer and length variables */
  hdr=(macd_header_t *) evt->buffer;
  len=evt->length;
  src=NTOI(hdr->src);  /* convert the network format to a 2-byte unsigned integer */
  dst=NTOI(hdr->dst);  /* convert the network format to a 2-byte unsigned integer */

  /* if the frame size is invalid - discard */
  if (len < MACD_HEADER_SIZE || len > cb->max_rx_size)
  {
    ++cb->mib.invalid_frame_count;
    (void)o_evt_free_chain(evt); /* discard the event */
  }
  /* if the protocol ID does not match - discard*/
  else if (hdr->protocol_id != MACD_PROTOCOL_ID)
  {
    ++cb->mib.invalid_protocol_id_count;
    (void)o_evt_free_chain(evt);
  }
  /* if the LAN ID does not match */
  else if (hdr->lan_id != cb->lan_id)
  {
    ++cb->mib.invalid_lan_id_count;
    if (MACD_IS_UNICAST(hdr))
    {
      lbt_set_reservation(hdr);
      (void)o_evt_free_chain(evt);
```

```
}
/* broadcast packets are posted to MAC-R if the LAN ID is
   0xff or if the masked LAN IDs match */
else if (dst==MACD _BROADCAST_ADDRESS)
{
  if (hdr->lan_id==MACD_BROADCAST_LAN_ID ||
     (hdr->lan_id & cb->lan_id_mask==cb->lan_id & cb->lan_id_mask))
  {
    /* post the buffer to the MAC-R layer. */
    MACR_receive_data_indication(cb, evt);
  }
  else
    (void)o_evt_free_chain(evt);
}
else /* multicast - but not broadcast */
  (void)o_evt_free_chain(evt);
}
/* else, if it is multicast */
else if (dst & 0x8000 || src & 0x8000)
{
  node_type=MACD_NODE_TYPE(dst);
  port_id=MACD_PORT_ID(dst);

  /* Is it addressed to an acceptable multicast address (i.e. for an AP)? */
  if ((node_type==O_NODE_ANY || node_type==O_NODE_AP) &&
      (port_id==O_PORT_ID_ANY ||
         port_id==MACD_PORT_ID(cb->address)))
  {
    /* post the buffer to the MAC-R layer. */
    MACR_receive_data_indication(cb, evt);
  }
  else
    (void)o_evt_free_chain(evt);
}
/* else, if it is unicast and addressed to me */
else if (dst==cb->address)
{
  if (MACD_IS_REQUEST_FRAME(hdr))
  {
    /* cancel the channel reservation if it is reserved by the
       source node */
    lbt_check_reservation(cb->lan_id,src);

    switch(cb->state)
    {
    case LISTEN:
      rx_idle(cb,evt);    /* rx_idle may change the high-level state to RECEIVE */
      break;

    case RECEIVE:
      rx_busy(cb, evt);
      break;

    case TRANSMIT:
```

```
      if (cb->tx_state.state==RDY_ENQ_E || cb->tx_state.state==S_ENQ_E)
        post_tx_done(ERROR_CLEAR_LOST);
      else
        transmit_abort(cb);  /* abort and retry later */
      cb->state=LISTEN;
      rx_idle(cb,evt);
      break;

    default:
      (void)o_evt_free_chain(evt);
      break;
    }

  }
  else /* poll frame */
  {
    switch (cb->state)
    {
    case TRANSMIT:
      if (src==cb->tx_state.dst)
        cb->tx_state.state(cb,evt);  /* pass the frame to the unicast transmit state machine */
      else
        (void)o_evt_free_chain(evt);
      break;

    case RECEIVE:
      (void)o_evt_free_chain(evt);
      break;

    default:
      (void)o_evt_free_chain(evt);
      break;
    }
  }
 }
 /* else, if it is unicast and not addressed to me */
 else
 {
   lbt_set_reservation(hdr);
   (void)o_evt_free_chain(evt);
 }
}
```

Appendix 2 - management variables.

MAC-D statistics.

The Direct Sequence/UHF MAC-D statistics are specific to the Direct Sequence/UHF radio cards. Media-independent statistics are kept in the MIB II interface group and Norand interface group on the AP. Note that frames which do not pass a CRC check are discarded and do not affect any MAC-D statistics.

Statistics which can be kept at the MAC-R layer:

MAC-D service time array - an array of counters which can be used to derive a histogram of MAC-D transmit request service times.

transmit request count - the total number of MAC-R transmit requests

transmit length errors - the total number of transmit requests which are rejected because the the maximum length was exceeded.

transmit request errors - the total number of rejected transmit requests.

access errors - the number of times that a transmit request failed because the channel was busy or reserved for the maximum number of retries.

RFP retry errors - the number of times that a transmit failed because the RFP retry count was exceeded.

ENQ retry errors - the number of times that a transmit failed because a POLL was lost (i.e. after max. ENQ retries).

lost CLEAR errors - the number of times that the status of a transmit request is unknown because the CLEAR was lost (i.e. after max. ENQ retries).

transmit timeout errors - the number of times that a transmit request failed because the time limit expired.

transmit abort errors - the number of times that a transmit request failed because the transmission was aborted (i.e. and restarted) too many times.

data retry errors - the number of times that a transmit request failed because a data fragment was retried the max. number of times.

tx protocol aborts - the number of times that a transmit request failed due to a protocol error.

Statistics which must be kept on the radio card:

tx frames - total number of transmitted frames.

tx data frames - total number of transmitted unicast DATA or EOD frames.

rx frames - total number of received frames.

rx data frames - total number of received unicast DATA or EOD frames.

rx overruns - the number of (i.e. data) frames discarded due to a lack of receive buffers.

channel access count - the number of times that the channel access algorithm was executed (i.e. before initiating a transmission).

channel busy count - the number of times that the channel was busy because RX detect was asserted.

channel reserved count - the number of times that the channel was reserved.

transmit abort count - the number of times that a transmission in progress was aborted.

invalid POLL count - the number of times that an invalid POLL or CLEAR frame was received, in response to a DATA or EOD frame.

receive timeout errors - the number of times that the receive state machine timed out (i.e. during the reception of a unicast packet).

invalid lan id count - the number of frames received with the wrong LAN ID.

rx access count - the number of times that the channel access algorithm was executed before responding to an RFP or ENQ frame.

rx busy count - the number of times that an RFP or ENQ was ignored because RX detect was asserted.

rx reserved count - the number of times that an RFP or ENQ was ignored because the channel was reserved.

rx restart count - the number of times that an RFP was received from the originator of the active conversation in the receive state (i.e. a DATA frame was followed by an RFP frame).

rx abort count - the number of times that an active conversation was aborted because an RFP frame was received from a node other than the originator.

reservation errors - the number of times that the originator or sink, of an active conversation, received a frame from some other node while the channel was reserved.

short frame count - the number of frames received which are too short.

long data frame count - the number of data frames received whichare too long.

protocol ID errors - the number of frames received which have an invalid protocol ID.

invalid frame count - the number of frames received with an invalid MAC-D header.

tx sequence errors - the number of POLL or CLEAR frames received with an invalid sequence number.

rx sequence errors - the number of out-of-sequence DATA or EOD frames received.

ENQ reject count - the number of times that a REJECT was sent in response to an ENQ frame.

DATA reject count - the number of times that a DATA or EOD frame was ignored because the reciever was not in the receive state.

MAC-D initialization variables.

- mode.
- channel.
- CSMA enable/disable.

Appendix 3 - master-slave coding examples.

The following code defines algorithms for calculating the total number of slots in an MSP response period and for calculating the response slot offset for a terminal.

```
#include <stdio.h>
#include <string.h>

typedef unsigned char uchar_t;
typedef unsigned short ushort_t;

/* 4 slots at level 0 */
#define MSP_L0_MASK  3
#define MSP_L0_SLOTS 4

/* default expansion factor for each level */
#define DEF_EXP_FACTOR 4

#define MSP_MAX_LEVEL 3
#define MSP_MAX_RSP_SLOTS 16

#define MSP_FLAG_RANDOM   0x01
#define MSP_FLAG_POLL_LIST 0x02

#define MSP_EXP_FACTOR(msp,lvl) \
((msp->exp_slots[0] & (0x8 >> lvl)) ? 4 : 2)

#define MSP_EXP_MASK(msp,lvl) \
((msp->exp_slots[0] & (0x8 >> lvl)) ? 3 : 1)

#define MSP_TEST_FLAG(flags,bit) (flags[bit>>3] & (0x80 >> (bit & 7)))
#define MSP_SET_FLAG(flags,bit) (flags[bit>>3] |= (0x80 >> (bit & 7)))

/*****************************************************************************
 * msp_expand_slot - recursively expands a branch of MSP expansion flags
 *****************************************************************************/
static unsigned msp_expand_slot(uchar_t *flags,
                    unsigned flag_cnt,
                    unsigned start_bit,
                    unsigned count,
                    unsigned *exp_factor,  /* 2 or 4 per level */
                    unsigned level,
                    slot_list_t *slp)
{
  unsigned slots=0;
  uchar_t *next_flags;
  unsigned next_flag_cnt;
  unsigned next_start_bit;
  unsigned index;
  unsigned i, test;

  next_flags = flags + (flag_cnt >> 3);
  next_flag_cnt = flag_cnt * exp_factor[level];  /* 2X or 4X */
```

```
next_start_bit = start_bit * exp_factor[level];

index = start_bit >> 3;         /* convert bit index to byte index */
test = 0x80 >> (start_bit & 7); /* point to the bit in the indexed byte */

for (i=0; i<count; i++)
{
  if (test & flags[index])
  {
    if (level==MSP_MAX_LEVEL-1)
    {
      slots+=exp_factor[MSP_MAX_LEVEL-1];
      if (slp)
        sl_add_slots(slp, exp_factor[MSP_MAX_LEVEL-1],
                 (uchar_t)MSP_MAX_LEVEL,
                 (uchar_t)next_start_bit);
    }
    else
      slots+=msp_expand_slot(next_flags,next_flag_cnt,
                     next_start_bit,exp_factor[level],
                     exp_factor,level+1,slp);
  }
  else
  {
    slots++;
    if (slp)
      sl_add_slots(slp,1,(uchar_t)level,(uchar_t)(start_bit+i));
  }

  test >>= 1;
  if (test==0)
  {
    test=0x80;
    ++index;
  }

  next_start_bit += exp_factor[level];
}

return slots;
}

/***************************************************************************
 * msp_calc_tot_slots - calculates the total slots for the "msp" and
 *   builds a slot list if "slp" is not NULL
 ***************************************************************************/
unsigned msp_calc_tot_slots(o_msp_t *msp, slot_list_t *slp)
{
  uchar_t *flags;
  unsigned start_bit;
  unsigned slots;
  unsigned test;
  int i;
  unsigned exp_factor[MSP_MAX_LEVEL+1];
```

```
  for (i=0; i <= MSP_MAX_LEVEL; i++)
    exp_factor[i]=MSP_EXP_FACTOR(msp,i);

  slots=0;
  flags=msp->exp_slots+1;
  start_bit=0;
  test=0x80;

  /* expand the level 0 slots */
  for (i=0; i < MSP_L0_SLOTS; i++)
  {
    if (test & msp->exp_slots[0])
    {
      slots += msp_expand_slot(flags, exp_factor[0] * MSP_L0_SLOTS,
                       start_bit, exp_factor[0],
                       exp_factor, 1, slp);
    }
    else
    {
      slots++;
      if (slp)
        sl_add_slots(slp,1,0,(uchar_t)i);
    }

    start_bit += exp_factor[0];
    test >>= 1;
  }

  return slots;
}

/****************************************************************************
 * node_slot_offset - calculates the response slot offset for a slave node
 ****************************************************************************/
int node_slot_offset(o_msp_t *msp, ushort_t node_id)
{
  uchar_t *flags;
  unsigned flag_cnt;
  unsigned start_bit;
  unsigned offset, i;
  unsigned slots;
  unsigned test;
  unsigned level;
  unsigned start_slot;
  unsigned exp_factor[MSP_MAX_LEVEL+1];
  unsigned exp_mask[MSP_MAX_LEVEL+1];

  start_slot=NTOI(msp->slot_offset) & 0xfff;

  for (i=0; i <= MSP_MAX_LEVEL; i++)
  {
    exp_factor[i]=MSP_EXP_FACTOR(msp,i);
    exp_mask[i]=MSP_EXP_MASK(msp,i);
  }
```

```
offset=node_id & MSP_L0_SLOTS-1;

slots=0;
flags=msp->exp_slots+1;
start_bit=0;
test=0x80;

/* expand previous slots */
for (i=0; i<offset; i++)
{
  if (test & msp->exp_slots[0])
  {
    /* recursive level 1..N expansion */
    slots += msp_expand_slot(flags, MSP_L0_SLOTS*exp_factor[0],
                    start_bit, exp_factor[0],
                    exp_factor, 1, NULL);
    start_bit += exp_factor[0];
  }
  else
    slots++;

  test >>= 1;
}

if (test & msp->exp_slots[0])
{
  /* level 1 expansion */
  start_bit=offset*exp_factor[0];
  node_id /= MSP_L0_SLOTS;
  offset=node_id & exp_mask[0];;

  flag_cnt=exp_factor[0] * MSP_L0_SLOTS;
  if (offset)
    slots += msp_expand_slot(flags, flag_cnt, start_bit,
                    offset, exp_factor, 1, NULL);

  /* level 2..N-1 expansion (level N cannot be expanded) */
  level=1;
  while (MSP_TEST_FLAG(flags,start_bit+offset) && level < MSP_MAX_LEVEL-1)
  {
    flags += flag_cnt >> 3;
    flag_cnt *= exp_factor[level];
    start_bit = (start_bit+offset) * exp_factor[level];
    node_id /= exp_factor[level-1];
    offset=node_id & exp_mask[level];
    ++level;
    slots+=msp_expand_slot(flags, flag_cnt, start_bit,
                    offset, exp_factor, level, NULL);
  }

  if (MSP_TEST_FLAG(flags,start_bit+offset))
  {
    node_id /= exp_factor[level-1];
```

```
      offset=node_id & exp_mask[level];
      slots += offset;
    }
  }

  start_slot=NTOI(msp->slot_offset) & 0xfff;

  return (int)(slots - start_slot);
}
```